

Distributed Virtual Microscopy and Microcharacterization (DVM^2)

B. Parvin

Working white paper

May 19, 1997

1 Introduction

The trend in control, communication, and computation continues toward an ever more intelligent man-machine interface achieved through operating instruments remotely, eliminating tedious off-line analysis, and improving instrument operation within the framework of open system for scalability. Scalability spans several dimensions and includes collaboratories. Here, the rationale is not just the economy, but also how an advanced instruments can become *equally* accessible and available through the Internet as well as how these instruments can be operated by users throughout the society of minds. Recent research [4] has indicated that collaboratories have the potential of closing the gap between professionals and organization, and to overcome the limitation of both space and time. The initial observation indicates i) a decrease in the amount of rework and subsequent analysis, ii) a positive change in the team dynamic when another discipline became the central problem, and iii) an increase in the number of alternative solutions for a given problem. Other researchers have made similar observations and proposed collaboratory tools and models of interaction [7]. Although significant progress has been made in the area of collaborative concurrent engineering and building design [2, 3, 12], our intent is to establish a set of *requirements* for implementing these trends for microscopy and microcharacterization (MM), and establish a framework –or framework of frameworks– for integrating currently funded DOE2000 R&D projects into the MMC collaboratory. Where a framework includes class libraries, APIs, and shared services. These requirements are driven in part by 1) functionality, 2) application interoperability, 3) scalability criterion, and 4) secure and safe operation of the microscope.

2 Requirements

The details of above requirements, their implementation and relationships to R&D programs are outlined below.

2.1 Functionality

The software platform should provide a functional model for operating the microscopes. The functional model shows how a particular value is computed and what the dependencies between different variables are. Functionality should include features such as query a particular microscope, parameters that can be controlled, predicting a particular change in the microscope state, verifying a potential change, and to allow communication in different coordinate systems.

2.2 Applications

The software environment should allow the users to conduct either static or dynamic (in-situ) experiments over the global Internet, and provide transparent access to the software analysis tools and legacy systems.

Dynamic experiments refer to those applications, where a specimen is externally stimulated through heat, stress, etc., and requiring real-time instrument control based on observed images. This is a task that is nearly impossible over the wide area network due to its insufficient bandwidth and unpredictable behavior. Agents will be utilized to provide the necessary automation and transparent behaviors.

An agent is a software utility that performs a specific task without human interaction. Hence, the user is transformed from a worker to a manager. In contrast, an intelligent agent can reason about its task and learn about its performance. Toward this end an intelligent agent is integrated, expressive, goal oriented, and customized. In the context of microscopy, two types of agents can be identified at this point. These include compensatory and notebook agents.

2.2.1 Compensatory agents

In many applications of microscopy, an operation or manipulation of a control parameter is followed by another one to ensure that the sample position remain stationary and visually acceptable. Three examples are provided here.

1. Tilting, at high magnification, should be followed by translation to keep an inclusion at the center of the field of view.
2. Translating a specimen may be followed by autofocusing to eliminate the 3D effects of the sample.
3. Stimulating an inclusion, i.e., heat, stress, etc., will cause drift and morphological changes in the inclusion. It is desirable to measure the drift, compensate for the drift, and quantify shape changes [9, 8] autonomously.

Microscopy is just one facet of scientific experiments. Other applications include comparison of physical models with the observed images, rendering and visualization of spectral images, compositional analysis, etc. For high-resolution electron microscopes, structural interpretation of images is facilitated by comparison with simulated HRTEM images. Such images should be available to the researcher on-line, at the microscope remote interface (GUI) and also for post-microscope comparison. These legacy systems also need to be supported as a part of any collaboratory framework.

2.2.2 Notebook agents

Any GUI can be programmed to timestamp user's commands and record them along with the observed data. Often a question may be "how did I get to this point?" And the system could playback various steps –some of which are meaningless.– However, a more interesting question would be to ask the differences, similarities, or generalization of protocols from different collaborators for observing a particular phenomenon for a given material.

Although notebook agents may seem far fetched for MMC applications, we will entertain their utilities and provide the necessary feedbacks to the notebook R&D project.

2.3 Scalability

Scalability for the software environment is a requirement that spans four dimensions, as shown in figure 1. These include i) the number of active users collaborating on a given experiment, ii) the number of different microscopes –from various vendors– that the software system can manipulate, iii) interoperability with other application programs, and iv) the variety platforms that the user interface can be executed on. Each of these requirements is now covered in more details. The intent is to have an architecture that is scalable across these four dimensions.

2.3.1 Number of active users

The software platform should allow many users to collaborate on a given experiment. In this environment, only one user will be operating and driving the microscope, while the other users will be observing. The current operating user will be able to hand off control to the next through an appropriate interlocking mechanism.

2.3.2 Number of microscopes

The software should be able to interface to any other microscope once the electronic interfaces have been put in place. For some of the newer microscopes, most of the

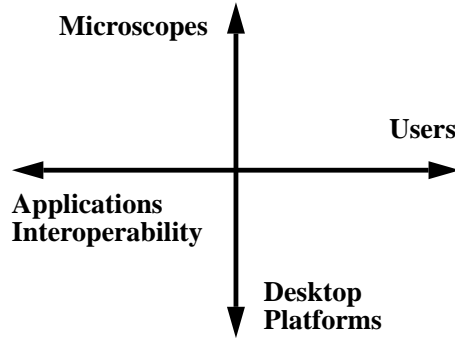


Figure 1: Dimensions of scalability

control functions are readily available through an external serial interface. However, each vendor has its own proprietary protocol that can ideally be expressed in terms of an appropriate API.

2.3.3 Application interoperability

Microscopy is just one facet of any scientific experiment. Subsequently, the data (images) need to be analyzed, stored, and processed by other application software that could be remotely located on another host. Hence, interoperability refers to smooth transition and interaction between various applications.

2.3.4 Number of platforms

Each remote user will be running one of the system clients that correspond to the graphical user interface. This GUI should be platform independent, i.e., Sun (Solaris), DEC (OSF), PC (Windows), and Macintosh, so that diverse users can participate.

2.4 Safety and Security

*DVM*² will extend the security protocol that is being developed by the R&D programs. This issue is addressed in Dr. Rome's white paper.

The instrument safety will be encoded through the user profile for maximum flexibility.

3 Software Architecture

DVM^2 will utilize extensible object oriented framework –class libraries, APIs, and shared services– so that applications can be rapidly assembled, maintained, and reused. Some of these utilities are being developed under the various R&D programs –funded by DOE2000– and they shall be evaluated and integrated as needed. DVM^2 should provide a functional model for microscopy and microcharacterization. However, since frameworks build on each other, the resulting applications are tightly linked through shared services infrastructure. The following service categories are envisioned for MMC as shown in figure 2.

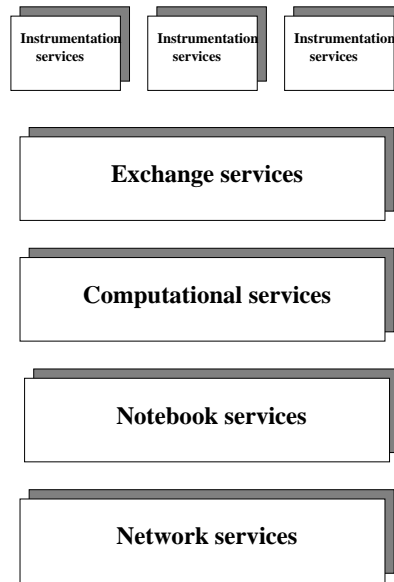


Figure 2: Layers of services for DVM^2

1. *Instrument services*: that are vertically integrated. Examples include TEM, SEM, AEM, etc.
2. *Exchange services*: that include generic processes and applications common to multiple types of instrument services. Examples include common GUI features, resource scheduling, etc.
3. *Computational services*: that allow for simulation, data mapping, real-time visual routines, visualization, etc.

4. *Notebook services*: that allow text, graphics, still images, and video to be stored and tabulated.
5. *Network services*: that enhance performance and reliability of the Internet, i.e., quality of service management, multicasting, authentication, and smart firewalls.

These core services could potentially be specified with an interface that is language independent. Some aspect of DVM^2 will be based on CORBA2 that includes Internet InterORB Protocol (IIOP), which Netscape Communicator will support. In CORBA, all objects connect to a common object bus as shown in figure 3. Thus, the actual implementation of figure 2 is flat. Some of the commercial work-flow systems are organized in this fashion.

From a user point of view, the system comprises of infrastructure, functionality, and computational toolkits. Infrastructure aims to meet the scalability requirements; functionality aims to provide the necessary tools for user's interaction, and application refers to the specific analysis tools that are needed for MM, i.e., image simulation, rendering, compositional analysis, basic diffraction analysis, Kikuchi analysis and Monte Carlo analysis. These objects may reside on any host and the architecture of figure 3 provides the necessary translator to bridge the gaps between various data formats. These objects can be listed, queried, and activated in the system.

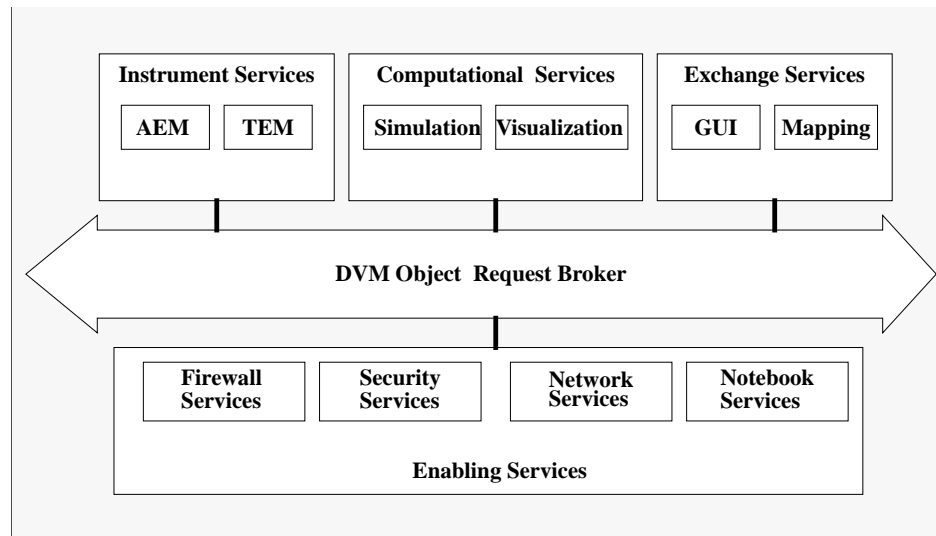


Figure 3: Interaction between enabling services and DVM^2 services

Some CORBA implementation supports the necessary scalability across three dimensions:

1. the number of active users through reliable –if necessary– multicasting from
 - (a) Vendors such as Isis Distributed Technologies and Olsen and Associates,
 - (b) Horus of Cornell University supporting a virtual synchronous execution model [1, 11], and
 - (c) DOE funded software bus project supporting secure reliable and unreliable multicasting, where security will be provided by another DOE funded project.
2. the number of various desktop platforms through automated marshalling and demarshalling of data; and
3. application interoperability for local/remote object location and activation.

The main shortcoming of CORBA is inefficient bulk data transfer across high speed networks such as ATM and FDDI [6, 5] and the lack of security context. For high speed data transfer, we shall use Adaptive Communication Environment (ACE) –a public domain software– that provides concurrent communication environment supporting both Solaris, and Windows NT platforms. ACE also supports unreliable multicasting, reactors, and uniform thread library, and it has been successfully deployed on the Spectrum and Irridium projects [10] in conjunction with CORBA. Furthermore, the DOE2000 security project is providing an ACE interface for their security protocol.

The software IOs are analogous to the graphic user interface (GUI) that runs on user's platform. The scalability for the GUI is accomplished through the Java language. The main benefits of Java are i) platform independent, ii) accessible through Web, iii) version control. Presently, each laboratory has implemented a JAVA based GUI to satisfy their own unique requirements. Here, a combination of CORBA, ACE, Java and corresponding JAVA Beans (an architecture and platform neutral API for creating and using dynamic JAVA components) provides three dimensions of the scalability axis to meet the reflective architectural requirements that were stated earlier. Another view of the overall system to be investigated is shown in figure 4.

3.1 Instrumentation services

The critical factor in scalability is the number of microscopes that the system needs to access. The natural choice is to have various vendors and manufacturers comply with a single standard. From a logistic point of view, this is a formidable and time

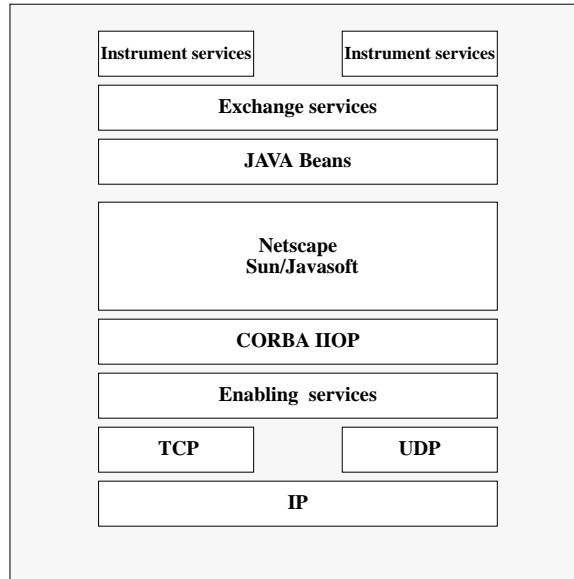


Figure 4: Layers of Interaction

consuming task. The current plan is to have each vendor –in the collaboratory– to provide an API that can be accessed through a commercial ORB in addition to the current script based protocol. These APIs will provide the necessary translators to their own special hardware. In the absence of vendor support, we will adopt an ad-hoc approach that will also be flexible once a standard is established. Here ad-hoc refers to polymorphism with respect to manipulating different microscopes, i.e., same interface but different drivers. The problem is further complicated by the fact that older or lower-cost EMs do not have the necessary computer controlled interfaces. Some of the industrial collaborators have developed the required hardware for retrofitting these microscopes. In addition, they plan to use some of the proposed software tools in their existing system. The basic computer control for an EM may include:

- four axis tilt and translation stage,
- filament control for automated saturation/desaturation of thermionic systems,
- tension control for automated conditioning,
- lens control for automated alignment and archival,
- STEM probe control for digital placement of the beam,

- aperture control for condenser, objective and selected area,
- specimen exchange, and
- vacuum monitoring system.

The ad-hoc approach is based on a *functional approach* to operation of the microscope. In this context, the user can interrogate how a particular value is computed, what are those values, what are the dependencies between pertinent values, etc. In this framework, two resource tables will be used at the implementation levels. These tables predefine a set of *components* and *constraints* on their properties and interactions. The component table lists each available control parameter and its corresponding range and domain, e.g., the range of possible magnifications. The constraint table lists the dependencies between various control parameters. If proposed values for parameters violate a constraint, a correction must be made. In this framework, the resource tables provide the translation mechanism for enforcing global consistency as defined by the parameters and constraints. These tables –for a pre-specified microscope– are propagated into the designated module for safe operation of the microscope. This approach is i) flexible for including a new microscope into the list of microscopes that can operated by the system; ii) hides all the details from the remote users; and iii) provides a degree of protection for safe operation of the microscope through the dependency table. The dependency table is not a necessary component of the system for professional users. However, it is meant to serve as an expert system to check manipulation of the system by novice users.

The functionalities of any microscope will be around properties and actions. For example, *focus* is a property. Properties can be a single variable or of array type. One possible abstraction of various properties is shown below¹. In this abstraction, an identical set of actions can be applied to every property.

¹Due to 1. John Hunt and Chris Meyer of Gatan, private communication, Aug. 1996, and 2. Hans de Ruijter of EMISPEC, private communication, Oct. 1996

Property	Action	Results
focus	cando	value
magnification	do	min
translation	set	max
tilt	get	structure
aperture		istable
beam position		table of n
diffraction		is valid
filament		description
probe		default value
image		
shutter		
vacuum		

4 Tasks

1. *Task 1:* The current production of LBNL insitu microscopy platform does not utilize object oriented methodologies. This baseline platform needs to be converted for software reusability and ease of maintenance. In addition, the next production version of LBNL baseline system will use low and high speed software busses.
2. *Task 2:* Development of efficient compression techniques for diffraction imaging and associated automation for alignment, development of an approach for quantitative analysis, compression, and visualization of spectral images.
3. *Task 3:* Integration of laboratory notebook, security tools for remote operation, and evaluating reliable multicasting.
4. *Task 4:* Design and implementation of a standardized data object vocabulary for accessing various types of microscopes.
5. *Task 5:* Design and development of a common JAVA based GUI for remote microscopy operations,
6. *Task 6:* Development of a session manager for graceful transition between remote operator and the local operator, or between users at different sites,
7. *Task 7:* Verification of a collaborative experiment to utilize unicasting for the remote operator while utilizing unreliable multicasting for the remote observers,
8. *Task 8:* Development of a framework for setting a layered degrees of trust for remote users,
9. *Task 9:* Integration and demonstration of above tasks on HVEM, Phillips, and a SEM at NCEM at LBNL.

5 Milestones

1. Year 1:

- T1: 0-6 month
- T2: 7-9, Develop and test efficient compression techniques for diffraction patterns,
- T3: 3-4, Identify data definition and data model for the laboratory notebook,
- T4: 0-3, Define a draft specification,
- T5: 2-4, a draft specification,
- T6: 7-11, a draft specification,
- T7: 7-11, Generate and evaluate test cases with the ACE,
- T8: 9-11, a draft specification,
- T9: 0-6, Hardware integration and testing for the CM200 and new functionalities to HVEM.

2. Year 2:

- T1: 12-23, Integrate additional features, refine, and test.
- T2: 12-15, Refine, validate, test, and incorporate user's feedback on compression of diffraction patterns. 12-24, Develop algorithms and software for analysis of spectrum images.
- T3: 12-16, Integration to electronic notebook and incorporation of user's feedback, and incorporation of point-to-point security architecture.
- T4: 12-20, Implementation of the draft specification and its integration to Task 1 followed by refinement and testing.
- T5: 12-15, Implementation and testing the draft GUI,
- T6: 17-20, Implementation of session manager followed by refinement.
- T7: 12-20, Implementation and integration to Task 1. Incorporation of user's feedback.
- T8: 21-23, Implementation of the draft document and its integration to task 1,
- T9: 12-23, bring CM200 on-line with LBNL baseline platform and realize full functionalities of HVEM.

3. Year 3:

- T1: 25-36, Incorporate user's feedback, document, and release.
- T2: 25-30 Test and refinement for spectrlimaging analysis.
- T3: 25-28 Incorporate user's feedback, document, and release.
- T4: 27-31 Test, refine, document and release.
- T5: 28-32 Test, refine, document and release.
- T6: 29-33 Test, refine, document and release.
- T7: 30-34 Test, refine, document and release.
- T8: 31-36 Test, refine, document and release.
- T9: 25-36 Test, refine, document and release.

Plan

- *Task 1:* The LBNL scientific requirements for in-situ microscopy is unique in that real-time control of a dynamic experiment is specified. One of the major objectives of task 1 will be to meet that goal. LBNL will support other laboratories with its agent software for general use.
- *Task 2:* The second tasks will be a joint collaboration between National labs and industry. LBNL will leverage its expertise to develop the required algorithms and software, and distribute them among the parties. LBNL will collaborate with ORNL to refine and integrate their auto-alignment techniques into their baseline system.
- *Task 3:* MMC will monitor DOE2000 R&D projects for integration into the MMC baseline remote microscopy platform.
- *Task 4:* This will be a joint cooperation between National labs and industries. A functional interface will be specified and integrated.
- *Task 5:* MMC will develop the required GUI and collect user's feedback from industry as well as other National labs.
- *Task 6:* The session manager for HVEM at LBNL has unique requirements and will build a system accordingly.
- *Task 7:* MMC will investigate communication libraries in ACE, CORBA and related DOE2000 R&D tasks to achieve this task.
- *Task 8:* MMC will work with industry partners and other National labs to develop a framework for safe operation of various instruments. Some of the implementations are unique to LBNL due to the unique instruments such as HVEM. Others, such as CM200, can benefit from shared cooperation.
- *Task 9:* MMC will test their baseline design on various systems at various National Laboratories.

References

- [1] K. Birman and R. Renesse. Software for reliable networks. *Scientific American*, 1996.
- [2] J. M. Clayton and et.al. Interpretation objects for multi-disciplinary design. In *AI in Design*, pages 573–590, 1994.
- [3] S.J. Fenves and et. al. Integrated software environment for building design and construction. *Computer-Aided Design*, 22:27–36, 1990.
- [4] R. Fruchter. Conceptual, collaborative building design through shared graphics. *IEEE Expert*, pages 33–41, 1996.
- [5] A. Gokale and D. Schmidt. Measuring the performance of communication middleware on high-speed networks. In *ACM SIGCOMM*, Stanford, 1996.
- [6] A. Gokale and D. Schmidt. Performance of the corba dynamic invocation interface and internet inter-orb protocol over high speed atm networks. In *GLOBE-COM*, London, 1996.
- [7] R. Kouzes, J. Myers, and W. Wulf. Collaboratories: Doing science on the internet. *IEEE Computer Magazine*, 29, 1996.
- [8] B. Parvin, D. Callahan, W. Johnston, and M. Measter. Visual servoing for micro manipulation. In *Int. Conference on Pattern Recognition*, Austria, 1996.
- [9] B. Parvin and et. al. Telepresence for in-situ microscopy. In *IEEE Int. Conference on Multimedia Systems and Computers*, Japan, 1996.
- [10] I. Pyarali, T. Harrison, and D. Schmidt. Design and performance of an object-oriented framework for high-speed electronic imaging. In *USENIX COOTS*, Toronto, Canada, 1996.
- [11] R. Renesse, K.P. Birman, and S. Maffeis. Horus, a flexible group communication system. *Communications of the ACM*, 1996.
- [12] D. Sriram and et. al. An object-oriented framework for collaborative engineering design. *Computer-Aided Cooperative Product Development*, pages 28–42, 1991.